ACTIVE MEMORY PROCESSING ARRAY TOPOGRAPHY AND METHOD

TECHNICAL FIELD

[001] The invention relates memory devices, and, more particularly, to a processing element array for an active memory device having a topography that minimizes the maximum length of the longest processing element interconnection path.

BACKGROUND OF THE INVENTION

[002] A common computer processing task involves sequentially processing large numbers of data items, such as data corresponding to each of a large number of pixels in an array. Processing data in this manner normally requires fetching each item of data from a memory device, performing a mathematical or logical calculation on that data, and then returning the processed data to the memory device. Performing such processing tasks at high speed is greatly facilitated by a high data bandwidth between the processor and the memory device. The data bandwidth between a processor and a memory device is proportional to the width of a data path between the processor and the memory device and the frequency at which the data are clocked between the processor and the memory device. Therefore, increasing either of these parameters will increase the data bandwidth between the processor and memory device, and hence the rate at which data can be processed.

[003] A memory device having its own processing resource is known as an active memory. Conventional active memory devices have been provided for mainframe computers in the form of discrete memory devices having dedicated processing resources. However, it is now possible to fabricate a memory device, particularly a dynamic random access memory ("DRAM") device, and one or more processors on a single integrated circuit chip. Single chip active memories have several advantageous properties. First, the data path between the DRAM device and the processor can be made very wide to provide a high data bandwidth between the DRAM device and the processor. In contrast, the data path between a discrete DRAM device and a processor is normally limited by constraints on

the size of external data buses. Further, because the DRAM device and the processor are on the same chip, the speed at which data can be clocked between the DRAM device and the processor can be relatively high, which also maximizes data bandwidth. The cost of an active memory fabricated on a single chip can is also less than the cost of a discrete memory device coupled to an external processor.

[004] An active memory device can be designed to operate at a very high speed by parallel processing data using a large number of processing elements ("PEs") each of which processes a respective group of the data bits. One type of parallel processor is known as a single instruction, multiple data ("SIMD") processor. In a SIMD processor, each of a large number of PEs simultaneously receive the same instructions, but they each process separate data. The instructions are generally provided to the PE's by a suitable device, such as a microprocessor. The advantages of SIMD processing are simple control, efficient use of available data bandwidth, and minimal logic hardware overhead. The number of PE's included on a single chip active memory can be very large, thereby resulting in a massively parallel processor capable of processing large amounts of data.

[005] A common operation in active memory SIMD processing element arrays is the shifting of an array of data. To facilitate this operation, processing elements in the array are preferably connected to each other by data paths that permit neighboring processing elements to transfer data between each other. For example, in a two-dimensional rectangular array, each processing element may be connected to its four nearest neighbors. To maximize the operating speed of the processing element array, it is desirable to minimizes the data path between processing elements. Doing so does not present a problem in the interior of the array because neighboring processing elements can be placed very close to each other. However, at the edges of the array, the path from a processing element at one edge of the array to a processing element at the opposite edge of the array (which are neighbors to each other) can be very large.

[006] One conventional technique that has been used to minimize the length of the longest path between processing elements can be explained with reference to Figures 1A-C, which show a technique for folding paper that can be analogized to "folding" an array of processing elements. As shown in Figure 1A, a rectangular piece of paper 10 representing a logical array of processing elements has edges 12, 14, 16, 18, a horizontal fold line 20, and two vertical fold lines 26, 28. The paper 10 is initially folded about the vertical fold lines 26, 28 as shown by the arrows 30, 32, 34, 36 in Figure 1A. After being folded in this manner, the paper has the configuration shown in Figure 1B. In this configuration, the vertical edges 12, 16 are positioned close to each other at the center of the paper 10. As a result, the distance between the vertical edges 12, 16 is relatively small, and the distances between the vertical edges 12, 16 and other portions of the paper 10 are also reduced.

[007] The paper 10 is next folded along the horizontal line 20 as shown by the arrows 40, 42, 44, 46 in Figure 1B, which results in the configuration shown in Figure 1C. In this configuration, the horizontal edges 18, 14 are positioned adjacent each other, as are the upper and lower portions of the vertical edges 12, 16.

[008] It is, of course, not possible to fold a semiconductor substrate on which processing elements are fabricated in the manner in which the paper 10 can be folded as shown in Figures 1A-C. However, a similar effect can be achieved by spacing processing elements apart from each other so that processing elements on "overlapping" portions of the substrate can be interleaved with each other. For example, with reference to Figure 1B, the processing elements between the vertical fold lines 26, 28 are spread out by one processing element. Processing elements in the substrate between the vertical fold line 26 and the vertical edge 16 and between the vertical fold line 28 and the vertical edge 12 are then positioned between the processing elements in the substrate between the fold lines 26, 28.

[009] When the substrate is "folded" as shown in Figure 1C, the processing elements are again interleaved. However, since there is now four layers of "substrate," the processing elements extending from one side of the folded substrate shown in Figure 1C must be interleaved by 3 processing elements. Since processing elements that are logically adjacent to each other before "folding" are now physically separated from each other by three processing elements, processing element interconnections are coupled to every fourth processing element.

[010] When an array of processing elements are conceptually "folded" as shown in Figure 1C, it can still be logically accessed as if it was in its unfolded configuration shown in Figure 1A. When folded as shown in Figure 1C, the array has physical topography shown in Figure 2, which, in the interest of clarity, shows only some of the processing elements in the array. Figure 2 also shows the physically location of registers 50, 52, 56, 58 that are positioned adjacent the edges 12, 14, 16, 18, respectively, of the array. When the array is folded about the horizontal line 20 as shown in Figure 1B, processing elements 60a-n logically positioned below the line 20 are physically to the left of processing elements 62a-n logically positioned above the line 20 as shown in Figure 2. The connections between the processing elements 62 logically positioned above the line and the processing elements 60 logically positioned below the line are physically connected to each other at the top, as shown in Figure 2. The processing element 60a at the logical bottom of the array and the processing element 62a at the logical top of the array are positioned adjacent a common edge register 68. However, a separate edge register may be provided for the processing elements 60 logically positioned below the line 20 and a separate edge register may be provided for the processing elements 62 logically positioned above the line 20. In either case, at least one edge register 68 is provided for each column of the array.

[011] As explained in greater detail below, Figure 2 also shows a first row of processing elements 70a-c,n logically below the line 20, a first row of processing elements 72n,n-1,a logically above the line 20, a second row of processing elements 74a-c,n logically above the line 20, and a second row of processing elements 76a-c logically above the line 20 (the processing element 62d is also labeled 74b, and the processing element 60d is also labeled 76n-1). However, the processing elements 70, 74 are actually in the same logical row above the line 20, and the processing elements 72, 76 are in the same logical row below the line 20. The processing elements 70 logically extend rightwardly from the left edge of the logical array, and the processing elements 72 logically extend leftwardly from the right edge of the logical array.

[012] A left edge register 80 physically positioned at the center of the physical array is logically positioned at the left edge of the logical array, *i.e.*, adjacent the line 16, three

processing elements above the line 20. A right edge register 82 is also physically positioned at the center of the substrate but is logically positioned at the right edge of the logical array, *i.e.*, adjacent the line 12. The left edge register 80 is coupled to a processing element 70a, which, in turn is coupled to a processing element 70b. The logical position of the processing element 70a is at the left edge of the logical array, *i.e.*, adjacent the line 16, three processing elements above the line 20, and the logical position of the processing element 70b is one processing element to the right of the left edge, three processing elements above the line 20. The processing elements to which the registers 80, 82 are coupled are in different logical rows.

[013] Similarly, the right edge register 82 is shown coupled to the processing elements 72a, n, n-1. The processing element 72a is one processing element to the left of the right edge of the logical array, three processing elements below the line 20. The processing element 72n is at the center of the logical array, and the processing element 72n-1 is to the right adjacent the processing element 72n, and both are logically positioned three processing elements below the horizontal line 20.

[014] The concept illustrated in Figure 1 and the topography shown in Figure 2 has the advantage of minimizing the length of the longest path between processing elements. However, the topography shown in Figure 2 has the disadvantage of making it difficult to perform operations using only a portion of an array of processing elements. For example, performing operations using only the processing elements logically positioned in the upper left quadrant shown in Figure 1 involves processing elements that are physically spread throughout the substrate. More specifically, the processing elements logically in the upper left quartile are interleaved with the processing elements in the lower left quartile, and they are in rows that are interleaved with processing elements in the upper and lower right quartiles.

[015] Therefore, a need exists for a processing array topography that minimizes the length of the longest path between processing elements in an array, but does so in a manner that facilities either partial or full use of the array.

SUMMARY OF THE INVENTION

[016] An array of processing elements, which may be included in an active memory device, are logically arranged in a rectangular grid of logical rows and logical columns so that which each processing element lies in only one logical row and one logical column. The processing elements in the array are divided into four sub-arrays each of which includes the processing elements in a respective quartile of the logical array. The processing elements in each of the sub-arrays are physically positioned in a folded arrangement. In this folded arrangement, the processing elements in different logical rows are physically interleaved with each other, and the processing elements in different logical columns are physically interleaved with each other. The processing elements in the array are coupled to each other by a system of conductors to allow data to be shifted from one processing element to a logically adjacent processing element.

BRIEF DESCRIPTION OF THE DRAWINGS

[017] Figures 1A-C are diagrams illustrating a prior art concept for minimizing the length of the longest path between processing elements in an array of processing elements.

[018] Figure 2 is a block diagram illustrating a topography for an array of processing elements using the concept illustrated in Figures 1A-C.

[019] Figures 3A-D are diagrams illustrating a topography for minimizing the length of the longest path between processing elements in an array of processing elements according to one embodiment of the invention.

[020] Figure 4 is a block diagram of an active memory device having an array of processing elements using the array topography illustrated in Figure 3D.

[021] Figure 5 is a block diagram of a computer system using the active memory device of Figure 4.

DETAILED DESCRIPTION OF THE INVENTION

[022] Figures 3A-D illustrate the concept of a topography for an array of processing elements according to one embodiment of the invention in the same manner that Figures

1A-C illustrate the prior art topography for an array of processing elements. As shown in Figure 3A, a piece of paper representing a logical array 90 of processing elements includes four edges 92, 94, 96, 98, one horizontal line 100 dividing the logical array 90 in two, two horizontal fold lines 102, 104, four vertical fold lines 110, 112, 114, 116, and a vertical divide line 120.

[023] With reference also to Figure 3B, the logical array 90 is severed into two array sections 130, 132 along the divide line 120, and each section 130, 132 is then individually folded. The logical array section 130 is folded about the vertical fold lines 110, 112 as shown by the arrows 140, 142, 144, 146 in Figure 3A, and the logical array section 132 is folded about the vertical fold lines 114, 116 as shown by the arrows 150, 152, 154, 156 in Figure 3A. Finally, each folded logical array section 130, 132 is folded as shown by the arrows 160, 162 and 164, 166, respectively, in Figure 3B to produce the configuration shown in Figure 3C. The array sections 130, 132 shown in Figure 3C are then positioned adjacent each other as shown in Figure 3D.

[024] Each of the array sections 130, 132 includes two sub-arrays 170, 172 and 180, 182, respectively, that have essentially the same topography as the topography shown in Figures 1A-C and 2. Significantly, each sub-array 170, 172, 180, 182 can operate independently because the processing elements in each of the sub-arrays 170, 172, 180, 182 are not interleaved with the processing elements in any of the other sub-arrays 170, 172, 180, 182. However, since the sub-arrays 170, 172 and 180, 182 in each of the array sections 130, 132, respectively, adjoin each other at the horizontal line 100, they can easily be interconnected to operate together. Further, since the array section 130 is positioned adjacent the array section 132, the sub-arrays 170, 172, 180, 182 can be interconnected with each other to operate together. Further, when the array sections 130, 132 are positioned as shown in Figure 3D, the upper and lower edges 94, 98 of the logical array 90 are all positioned closely adjacent each other. As a result, the length of the paths between processing elements both within each sub-array 170, 172, 180, 182 and between the sub-arrays 170, 172, 180, 182 are relatively short. In one embodiment of the invention, the logical array 90 includes 1024 processing elements so that each sub-array 170, 172, 180, 182 includes 256 processing elements.

[025] Figure 4 shows an active memory device 200 using a processing array topography according to one embodiment of the invention. The memory device 200 is coupled to a host 214, such as a microprocessor, although it may be coupled to other devices that supply high level instructions to the memory device 200. The memory device 200 includes a first in, first out ("FIFO") buffer 218 that receives high level tasks from the host 214. Each task includes a task command and may include a task address. The received task commands are buffered by the FIFO buffer 218 and passed to a command engine 220 at the proper time and in the order in which they are received. The command engine 220 generates respective sequences of commands corresponding to received task commands. These commands are at a lower level than the task commands received by the command engine 220. The commands are coupled from the command engine 220 to either a first FIFO buffer 224 or a second FIFO buffer 228 depending upon whether the commands are array processing commands or memory commands. If the commands are array processing commands, they are passed to the FIFO buffer 224 and then to a processing array control unit ("ACU") 230. If the commands are memory commands, they are passed to the FIFO buffer 228 and then to a DRAM Control Unit ("DCU") 234.

[026] The ACU 230 executes an intrinsic routine containing several microinstructions responsive to each command from the FIFO buffer 224, and these microinstructions are executed by an array of PEs 240. The PEs preferably operate as SIMD processors in which all of the PEs 240 receive and simultaneously execute the same instructions, but they do so on different data or operands. However, the PEs may also operate as multiple instruction, multiple data ("MIMD") processors or some other type of processors. In the embodiment shown in Figure 4, there are 1024 PEs 240 arranged in 4 sub-arrays of 256 PEs 240 each using the topography illustrated in Figure 3D. Each of the PEs 240 is coupled to receive 8 bits of data from the DRAM 244 through register files 246. In the embodiment shown in Figure 4, the DRAM 244 stores 16M bytes of data. However, it should be understood that the number of PEs 240 used in the active memory device 200 can be greater or lesser than

1024, and the storage capacity of the DRAM 244 can be greater or lesser than 16 Mbytes. Each of the sub-arrays 170, 172, 180, 182 preferably interfaces with its own register file 246 and its own interface to the DRAM 244. Although not shown in Figure 4, each processing array section 130, 132 preferably has its own DRAM 244.

[027] In operation, different intrinsic routines containing different microinstructions are issued by the ACU 230 for different commands received from the FIFO buffer 224. The DCU 234 issues memory commands and addresses responsive to commands from the FIFO buffer 224. In response, data are either read from a DRAM 244 and transferred to the register files 246, or written to the DRAM 244 from the register files 246. The register files 246 are also available to the PEs 240. The ACU 230 and the DCU 234 are coupled to each other so the operation of each of them can be synchronized to the other. The ACU 230 and DCU 234 are also coupled directly to the register files 246 so that they can control the operation and timing of data transfers between the register files 246 and both the PEs 240 and the DRAM 244.

[028] With further reference to Figure 4, the DRAM 244 may also be accessed by the host 214 directly through a host/memory interface ("HMI") port 248. The HMI port 248 receives commands that are substantially similar to the commands received by a conventional SDRAM except that signals for performing a "handshaking" function with the host 214 may also be provided. These commands include, for example, ACTIVE, DEACTIVATE, READ, WRITE, etc. In the embodiment shown in Figure 4, the HMI port 248 includes a 32-bit data bus and a 14-bit address bus, which is capable of addressing 16,384 pages of 256 words. The address mapping mode is configurable to allow data to be accessed as 8, 16 or 32 bit words.

[029] In a typical processing task, data read from the DRAM 244 are stored in the register files 246. The data stored in the register files 246 are then transferred to the PEs 240 where they become one or more operands for processing by the PEs 240. Groups of data bits read from or written to each set of DRAM columns are processed by respective PEs 240. The data resulting from the processing are then transferred from the PEs 240 and stored in the register files 246. Finally, the results data stored in the register files 246 are written to the DRAM 244.

[030] The PEs 240 operate in synchronism with a processor clock signal (not shown in Figure 4). The number of processor clock cycles required to perform a task will depend upon the nature of the task and the number of operands that must be fetched and then stored to complete the task. In the embodiment of Figure 4, DRAM operations, such as writing data to and reading data from the DRAM 244, requires about 16 processor clock cycles. Therefore, for example, if a task requires transferring three operands into and of the DRAM 244, the task will require a minimum of 48 cycles.

[031] A computer system 300 using the active memory device 200 of Figure 4 or some other active memory device having a processing element array topography according to the present invention is shown in Figure 5. The computer system 300 includes a processor 302 for performing various computing functions, such as executing specific software to perform specific calculations or tasks. The processor 302 includes a processor bus 304 that normally includes an address bus, a control bus, and a data bus. In addition, the computer system 300 includes one or more input devices 314, such as a keyboard or a mouse, coupled to the processor 302 through a system controller 310 to allow an operator to interface with the computer system 300. Typically, the computer system 300 also includes one or more output devices 316 coupled to the processor 302 through the system controller 310, such output devices typically being a printer or a video terminal. One or more data storage devices 318 are also typically coupled to the processor 302 through the system controller 310 to store data or retrieve data from external storage media (not shown). Examples of typical storage devices 318 include hard and floppy disks, tape cassettes, and compact disk read-only memories (CD-ROMs). The processor 302 is also typically coupled to a cache memory 326, which is usually static random access memory ("SRAM"). The processor 302 is also coupled through the data bus of the processor bus 304 to the active memory device 200 so that the processor 302 can act as a host 214, as explained above with reference to Figure 4.

[032] From the foregoing it will be appreciated that, although specific embodiments of the invention have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.